Application For United States Patent

For

INBOUND PACKET PLACEMENT IN HOST MEMORY

By

Avigdor Eldar and Fabian Trumper

Attorney Docket No: P18438

Firm No. 77.0070

David Victor, Reg. No. 39,867
KONRAD RAYNES & VICTOR, LLP
315 So. Beverly Dr., Ste. 210
Beverly Hills, California 90212
(310) 556-7983

INBOUND PACKET PLACEMENT IN HOST MEMORY

BACKGROUND

[0001] Adapters that transmit data over a network using the Transmission Control Protocol (TCP)/Internet Protocol (IP) communication protocol write the packets to host memory where the packets are processed by a host system software driver. Adapters using network communication protocols other than TCP/IP may also copy inbound traffic to the host memory. Further details of the TCP/IP protocol are described in the publications "A TCP/IP Tutorial", Request for Comment No. 1180, published by the Internet Engineering Task Force (IETF) (Jan. 1991) and "Requirements for Internet Hosts -- Communication Layers", RFC No. 1122, published by the IETF (Oct. 1989). Network adapters may also transmit packets using additional protocols known in the art, such as Layer 2 packets, like Ethernet & Token Ring, which encompass TCP packets. Layer II protocols are defined in IEEE 802.3, 802.5, 802.11, 802.14 Specifications. When writing the packets, the adapter places information on a packet to write to host memory in a descriptor data structure that references a host memory buffer of a fixed byte size, such as 1514 bytes, in which one packet is placed. Each received packet is written to one or more buffers in memory that is referenced by a separate descriptor. The adapter adds an updated descriptor including information on the packet in the memory buffer referenced by the descriptor to a descriptor table or array that the host software driver uses to access the packets. When the packet is in memory, the adapter hardware signals the software driver that a packet is available in memory and ready to process.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates a computing environment;

FIG. 2 illustrates descriptor information;

FIG. 3 illustrates how descriptor array entries reference buffers including packets;

FIG. 4 illustrates operations to transfer packets to host memory buffers; and

FIG. 5 illustrates operations performed by the adapter device driver to access packets in the host memory.

DETAILED DESCRIPTION

[0003] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several embodiments. It is understood that other embodiments may be utilized and structural and operational changes may be made without departing from the scope of the embodiments.

[0004] FIG. 1 illustrates a computing environment used with the described embodiments. A host system 2 has a processor 4, which may comprise one or more central processing units (CPU), a memory 6, an operating system 8, and is coupled to an adapter 10, such as a network interface card (NIC), to communicate over a network 12. The adapter 10 may be mounted on the host 2 motherboard, such as a Local Area Network (LAN) on Motherboard implementation, or be implemented in an expansion card that is inserted in a slot on the host 2 motherboard. An adapter device driver 14 executes in the memory 6 to provide an interface between the operating system 8 and the adapter 10 and performs such operations as managing interrupts, making device calls to control the adapter 10, and transmitting packets to the adapter 10. The adapter 10 and device driver 14 communicate over a bus interface 11, which may comprise bus interfaces known in the art, such as the Peripheral Component Interconnect (PCI) bus described in the publication "Conventional PCI 2.3", published by the PCI-SIG. There may be additional adapters in the host system.

[0005] The adapter 10 includes a packet reception engine 20 to process incoming packets. The packet reception engine 20 may implement network layer protocols, e.g., Ethernet, to process the received packets. Details of the Ethernet protocol are described in IEEE 802.3. In certain embodiments, certain of the TCP/IP protocol related operations may be offloaded into the packet reception engine 20 in the adapter 10. The adapter 10 further includes a network layer protocol for implementing the physical communication layer to send and receive network packets to and from remote devices over the network 12. Upon receiving network packets, the adapter 10 builds receive descriptor entries that are added to a receive descriptor array 22 accessible to both the adapter 10 hardware and the adapter driver 14 software executing in the host 2. The adapter 10 may include a Direct Memory Access (DMA) engine to build and update descriptors and transmit data to host memory 6. After creating receive descriptor entries, the adapter 10 may DMA the data (packet) it received from the network into a packet buffer, e.g., 26, in the packet buffer 28 area of the host memory 6 at the buffer address. The descriptor entries in the descriptor array 22 indicates to the adapter 10 hardware where to DMA the data and indicates to the adapter device driver 14 where to access the packet to return to a protocol driver, such as a TCP/IP protocol driver for processing the coded packets. In certain embodiments, each packet buffer, e.g., 26, may comprise a memory page which may include multiple packets.

[0006] FIG. 2 illustrates information included in a descriptor entry 50 included in the receive descriptor array 22. In certain embodiments, each descriptor entry 50 is associated with one packet. The descriptor entry 50 includes a descriptor identifier 52 providing an identifier of the descriptor, a packet length 54 indicating the length of the packet, a buffer address 56 providing the physical address of the start of the buffer, e.g., 26, and a number of packets placed 58 indicating the number of packets contiguously stored in the buffer 26 referenced by the descriptor entry 50. In certain embodiments, the descriptor entry 50 referencing the first packet in the buffer, e.g., 26, includes the buffer address indicating the start of the buffer, but the descriptor entries for all subsequent packets in that buffer may not include the buffer address. In certain embodiments, the descriptor entries for the subsequent packets in a buffer may indicate the descriptor of the first packet or the buffer address in which they are included. In still alternative embodiments, the descriptors referencing the subsequent packets following the first packets in a packet buffer, e.g., 26, may be determined as the descriptor entries following the descriptor entry referencing the first packet. The number of descriptor entries referencing subsequent packets following the first packet that are in the buffer 26 are determined from the number of packets in the packet buffers 28. The descriptor entry 50 may include additional information 60, such as the status of the adapter 10 DMA operation to transfer the packets from the adapter 10 to the packet buffer 26, the IP checksum status, etc.

[0007] FIG. 3 illustrates an example of how entries in the descriptor array 22 may reference packets in packet buffers 28. For instance, descriptor 70a references a first packet 72a in packet buffer 74a. Descriptors 70b and 70c reference the subsequent packet frames 72b and 72c in packet buffer 74a. Descriptor 70d references packet frame 72d in buffer 74b and descriptor 70e references packet frame 72e in buffer 74c.

[0008] FIG. 4 illustrates operations performed by the packet reception engine 20 to store packets in the packet buffers 28 (FIG. 1). A first packet is received (at block 100) for a buffer, e.g., 26, in memory 6. A descriptor 50 is generated (at block 102) indicating a length of the first packet and a buffer address of the buffer. The length of the packet may be indicated in the packet length field 54 and the buffer address indicated in field 56. Upon receiving a subsequent packet (at block 104), the packet reception engine 20 generates (at block 106) a descriptor 50 for the subsequent packet indicating a length of the packet, such as in the packet length 54 field of the new descriptor. If (at block 108) the current buffer 26 being used has available space for the first packet and the at least one subsequent packet received before transferring the packets to the buffer, then control returns to block 104 to receive a further packet. Thus, in certain embodiments, the packet reception engine 20 delays writing a group of received one or more packets if the packets received so far can fit into the currently used buffer 26 with space to spare for any further packets. In this way, all the packets that are intended to fit into one packet buffer 26 may be transferred to the host memory 6 in a single bus 11 transaction. In alternative embodiments, the adapter 10 may send the packets for a single buffer in multiple bus transactions.

[0009] If (at block 108) there is no available space in the current buffer 26 for the recently received packet and previously received packets not yet transferred, then operations are performed to transfer to the buffer the first packet and any intervening packets between the first packet and recently received subsequent packets. In alternative embodiments, the packets may be transferred if the available space is less than a minimum possible packet size, which may include the size of headers. If there is not sufficient space, then indication is made (at block 110) in the descriptor 50 for the first packet of a number of packets 50 that will be transferred to the buffer 26, including the first packet and any subsequent packets transferred to the buffer with the first packet, which may not include the recently received subsequent packet. The descriptors 50 for the first packet and the at least one subsequent packet written to the buffer with the first descriptor are written (at block 112) to the descriptor array 22.

[0010] Further, the descriptor the packet reception engine 20 generates (at block 114) for the received subsequent packet indicates a next buffer address of a next available buffer in the memory 6, e.g., 74b (FIG. 3). In such case, the subsequent packet., e.g., 72d, becomes the first packet for the next available buffer, e.g., 74b. The first packet and any intervening packets received between the first packet and the received subsequent packet, e.g., 72a, 72b, 72c, are transferred (at block 116) in response to receiving the subsequent packet that does not fit into the buffer with the first packet and the any intervening packets. In certain embodiments, the first packet and each subsequent packet is transmitted on a bus 11 into the buffer 26 in a single bus transaction to the buffer.

[0011] FIG. 5 illustrates operations performed by the adapter device driver 14 to use the receive descriptor array 22 to process packets placed in the packet buffer 28 by the packet reception engine 20. In certain embodiments, the packet reception engine 20 signals the adapter device driver 14 to process packets in the packet buffers 28 after transferring the packets to the packet buffers 28. At block 150, the adapter device driver 14 accesses a first descriptor 50 referencing a first packet, e.g., 72a (FIG. 3), in a buffer, e.g., 74a, and indicating a number of packets in the buffer, such as in field 58 (FIG. 2). A first packet, e.g., 72a, is extracted (at block 152) from the buffer, e.g., 74a, identified in the first descriptor, such as in buffer address field 56. At block 154, at least one subsequent descriptor, e.g., 70b, 70c, 70d, 70e (FIG. 3), in the descriptor array 22 is accessed. The accessed descriptor is associated with one subsequent packet indicated in the number of packets and the subsequent descriptor indicates a length of the associated subsequent packet. Each accessed subsequent descriptor is used (at block 156) to access the associated subsequent packet, e.g., 72b, 72c, in the buffer, e.g., 74a. The descriptors, e.g., 70a, 70b, 70c, are released after extracting the packets, e.g., 72a, 72b, 72c associated with the descriptors.

[0012] By allowing multiple packets to share a memory buffer, described embodiments optimize packet processing for small packets. Certain embodiments, such as streaming media or voice over Internet Protocol (IP) transfer relatively small sized packets, such as between 64 to128 bytes. The described embodiments allow multiple of such smaller packets to be stored in a single memory buffer, which may comprise the maximum transfer unit for Ethernet, e.g., 1514 bytes, to optimize space utilization of the packet buffers. Further, in embodiments where all packets for a memory buffer are transferred in a same bus transaction, bus overhead is conserved because the overhead per transaction is reduced by placing multiple packets into single bus transactions.

[0013] Further, placing multiple packets into a single memory buffer improves processor cache utilization. For instance, many processors use on-board cache, and load data into cache lines of the on-board cache, where each cache line may have space for 64 or 128 byte lines. With the described embodiments, if the content of a buffer is loaded into the processor cache, then the processor cache will comprise mostly packet data because, with the described embodiments, as much free space as possible in the buffer is utilized to store packets, minimizing the use of cache lines to store empty space in the buffers. Further, the number of cache hits is increased by including multiple packets into a single cache line to maximize the number of packets subject to cache requests in the packet buffer space. One additional reason for increased cached hits is cache-line prefetching: accessing cache-line triggers in an early fetch (pre-fetch) of cache-line X+1. Thus, if a number of small packets reside in consecutive cache-lines, packets will be ready in CPU cache when the processor performs data access operations.

[0014] Yet further, in embodiments where each buffer is implemented in a separate memory page, the overhead associated with accessing memory pages is minimized by including multiple packets in a single buffer/memory page. This reduces the number of different memory pages that are accessed in order to access multiple packets.

Additional Embodiment Details

[0015] The described embodiments may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" as used herein refers to code or logic implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.) or a computer readable medium, such as magnetic storage medium (e.g., hard disk drives, floppy disks,, tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which preferred embodiments are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Thus, the "article of manufacture" may comprise the medium in which the code is embodied. Additionally, the "article of manufacture" may comprise a combination of hardware and software components in which the code is embodied, processed, and executed. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the embodiments, and that the article of manufacture may comprise any information bearing medium known in the art.

[0016] The described operations may be performed by circuitry, where "circuitry" refers to either hardware or software or a combination thereof. The circuitry for performing the operations of the described embodiments may comprise a hardware device, such as an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc. The circuitry may also comprise a processor component, such as an integrated circuit, and code in a computer readable medium, such as memory, wherein the code is executed by the processor to perform the operations of the described embodiments.

[0017] FIG. 2 illustrates an example of information included in a descriptor. The descriptor information may be stored in a different format than shown in FIG. 2 with additional or less information on each connection between two devices and the information on the devices.

[0018] The described embodiments concern the transmittal of packets from a network adapter to host memory. However, the described embodiments for buffering packets may apply to packets received from Input/Output devices other than network adapters, such as a storage interface, printer interface, etc.

[0019] In described embodiments, packets were transmitted to a buffer after a point was reached that no further packets could fit into the buffer. In alternative embodiments, packets may be transmitted to a buffer even if the buffer has available free space for further packets. For instance, the packets may be transmitted to a single buffer if a timer has expired, where the timer is reset when starting a new buffer to receive packets. The adapter may also send the packets if the adapter does not want to bundle consecutive packets. The adapter may only bundle packets in a single buffer that are going to be processed by a same host processor, so that different buffers may have packets to be processed by different host processors.

[0020] The illustrated operations of FIGs. 4 and 5 show certain events occurring in a certain order. In alternative embodiments, certain operations may be performed in a different order, modified or removed. Moreover, operations may be added to the above described logic and still conform to the described embodiments. Further, operations described herein may occur sequentially or certain operations may be processed in parallel. Yet further, operations may be performed by a single processing unit or by distributed processing units.

[0021] The foregoing description of various embodiments has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the embodiments to the precise form disclosed. Many modifications and variations are possible in light of the above teaching.